Ad-Hoc Federated Learning on Edge Devices

Nikhil Kabra
nk23343@eid.utexas.edu

Ronak Jain
ronakjain0919@utexas.edu

Abstract

This project explores how Federated Learning can remain resilient under unreliable network conditions. Using a centralized FedAvg framework with configurable network topologies, we establish a working simulation that tracks accuracy, and communication metrics, forming the structure for introducing disturbances and adaptive recovery mechanisms.

1. Introduction

Federated Learning (FL) enables collaborative model training across distributed devices while preserving data privacy by sharing only model updates with a central server. However, in real-world wireless networks, FL performance is often hindered by unreliable edge devices with intermittent connectivity, and limited bandwidth, which can disrupt training and slow model convergence [1].

To address these challenges, this project aims to improve the robustness of FL in ad-hoc networks. Using a standard FedAvg framework, we simulate realistic network disturbances and design dynamic re-routing algorithms that adapt to device failures, with the goal of maintaining model accuracy and stability under real-world edge conditions.

2. Previous Work

Federated Learning (FL) preserves data privacy but traditionally relies on a central server, creating a single point of failure and communication bottlenecks [1]. To mitigate this, recent works have shifted toward decentralized architectures. Khan et al. [2] introduced socially aware clustering to optimize device-to-device communication based on stable social ties, while Kim et al. [3] developed dynamic clustering to address statistical heterogeneity (non-IID data) and improve convergence speed.

However, these frameworks exhibit critical gaps regarding routing resilience and disturbance models. Khan et al. [2] rely on static metrics to form clusters, assuming that once a link is established, it remains stable. Similarly, Kim et al. [3] adjust clusters for data distribution but do not account for physical network reliability. Standard industry protocols, such as those described by Bonawitz et al. [4], handle network disturbances by simply treating disconnected devices as "stragglers" and dropping them from the round. This approach wastes partial computations and degrades performance in unstable edge environments.

Our project fills this specific gap by introducing dynamic re-routing protocols for fault recovery. Unlike the static transmission paths in [2] or the drop-out mechanisms in [4], our framework actively re-routes updates through neighboring peers upon detecting a link disturbance. We evaluate this on the MNIST dataset, demonstrating that our adaptive routing maintains accuracy even when subjected to stochastic network disruptions that cause existing methods to fail.

3. Approach

System Modeling and Network Assumptions To simulate a realistic ad hoc edge environment, the code constructs a dynamic network where devices function as nodes in a configurable graph. We utilize NetworkX to generate these topologies, employing Erdős–Rényi and scale-free structures to model changing connectivity. The simulation assumes that each device operates under specific physical constraints; consequently, every node is assigned attributes including bandwidth, latency, and online availability. To reflect real-world statistical heterogeneity, the system partitions datasets across these clients using both IID and non-IID splits. Unlike earlier milestones that assumed a centralized, fault-free connection, this iteration activates a full communication layer. This requires model updates to traverse multi-hop paths, preserving synchronous rounds while adhering to strict rules regarding late updates.

Federated Learning Tasks and Execution: The core learning process relies on the FedAvg algorithm, but with modifications to support personalization and ad hoc transport. The execution flow begins with local training, where each device updates its model on its local data partition. To ensure consistent convergence behavior across experiments, we utilize a fixed learning rate of 0.01 and local batch size to 64. To handle diverse data distributions without synchronization issues, we incorporate Federated Batch Normalization. Once training is complete, the code initiates the aggregation phase: the server collects updates, computes a weighted average based on the number of samples per device, and broadcasts the global model back to the network. The implementation parallelizes local training and utilizes vectorized operations in NumPy or PyTorch for aggregation, allowing the system to scale efficiently to networks of hundreds or thousands of devices. The software architecture is modular, strictly separating data loading, training loops, and disturbance handling to ensure reproducibility.

Naive Approach: The naive routing approach finds paths using the shortest-hop BFS over the original static graph. It selects the path with the fewest hops to any gateway, then checks if it contains offline nodes or failed links—if it does, it fails immediately without trying alternatives. It estimates transmission time by multiplying hop count by a fixed 0.1 seconds per hop, ignoring link bandwidth, latency, and payload size. This makes it simple but brittle: any failure along the shortest-hop path causes delivery to fail, and it doesn't adapt to network conditions or optimize actual transmission time like the dynamic approach does.

Disturbance Models and Routing Recovery: To evaluate resilience, the system actively injects stochastic disturbances parameterized by specific failure probabilities. We model network instability by applying a link failure probability of 0.27, and a device dropout probability of 0.37. When a device cannot directly communicate with the server due to a disturbance, the code does not immediately discard the update. Instead, it identifies a set of high-degree nodes to act as gateways and executes Dijkstra's algorithm on the active graph to calculate an alternative route. The feasibility of this route is determined by the cost function which integrates latency and bandwidth constraints. The cost function is defined as the following:

(1)

If a valid path to a gateway exists within the communication budget, the update is re-routed; if the path is too costly or the device is depleted, the update is treated as a terminal failure and discarded. This logic allows the system to degrade gracefully, prioritizing recovery via multi-hop routing whenever physically possible. The pseudocode is described in Figure 1.

Figure 1: Pseudocode for Disturbances and Recovery Algorithm

4. Experimental Setup and Results

Data Distribution: To ensure a rigorous evaluation of the proposed framework, we simulated a Federated Learning environment using the EMNIST dataset distributed across 100 nodes. To replicate realistic edge heterogeneity, we implemented a non-IID data split using a class-wise Dirichlet distribution. We utilized a concentration parameter of 0.5 to generate moderate data skew, ensuring that local models diverge significantly without rendering training impossible.

Hyperparameters and Optimization: For M3, we optimized the training process to improve stability under these non-IID conditions. The system utilizes an increased batch size of 64, a decreased learning rate of 0.01, and increases participation to 20 clients per round. Crucially, we introduced adaptive sampling, which proactively excludes nodes with critically low battery levels from selection. This optimization is significant because it prevents mid-round dropouts that would otherwise waste computation cycles and destabilize the global aggregate, ensuring that selected clients are physically capable of completing the training task.

Results for Erdős–Rényi (ER) Topology: We first evaluated performance on an Erdős–Rényi (ER) graph, representing a random network structure. The performance metrics for the three test scenarios are detailed below:

- **Ideal Scenario**: Achieved an accuracy of 80.25%. This is shown in Figure 2.
- **Naive Approach**: Performance dropped significantly to 62.04%. This is shown in Figure 3
- **Recovery Protocol**: Our approach achieved 77.69%, virtually matching the Ideal scenario. This is shown in Figure 4.

This result validates that our dynamic re-routing algorithm effectively negates the impact of link failures in random graphs. By re-routing updates through neighbors, the system preserves the statistical diversity of the training data.

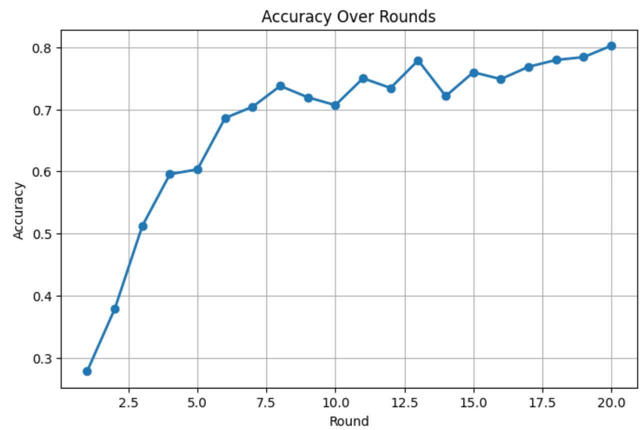

Figure 2: Ideal Scenario for Random Networks

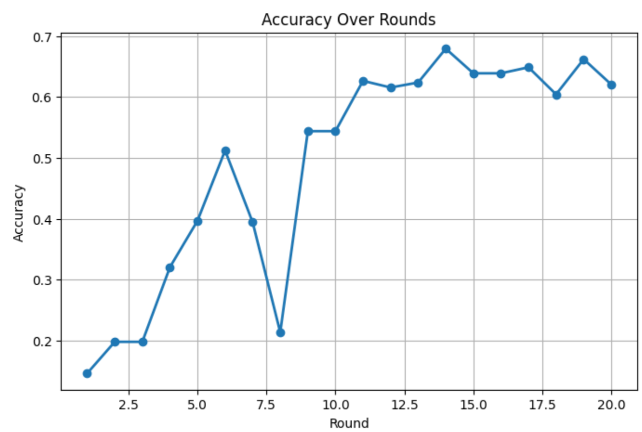

Figure 3: Naive Approach for Random Networks

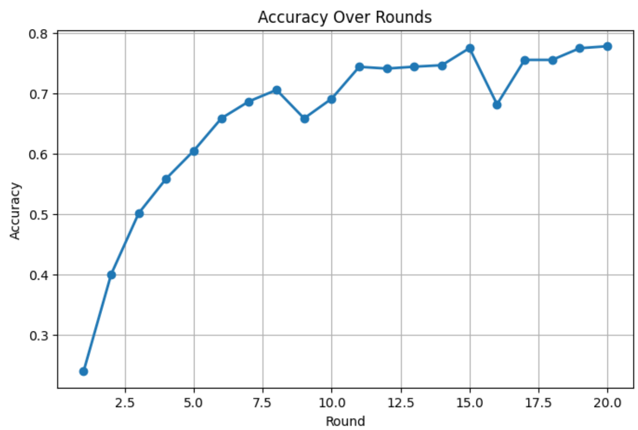

Figure 4: Recovery Scenario for Random Network

Results for Scale-Free Topology: We extended the evaluation to a Scale-Free topology to simulate networks with power-law degree distributions, such as social networks or the internet, where "hub" nodes exist.

- Ideal Scenario: Achieved an accuracy of 78.33%, which is highly like the ER baseline. This is shown in Figure 5.
- Naive Approach: Performance dropped significantly to 54.69%. This is shown in Figure 6.
- Recovery Protocol: 77.17% accuracy which is close to the Ideal Scenario. This is shown in Figure 7.

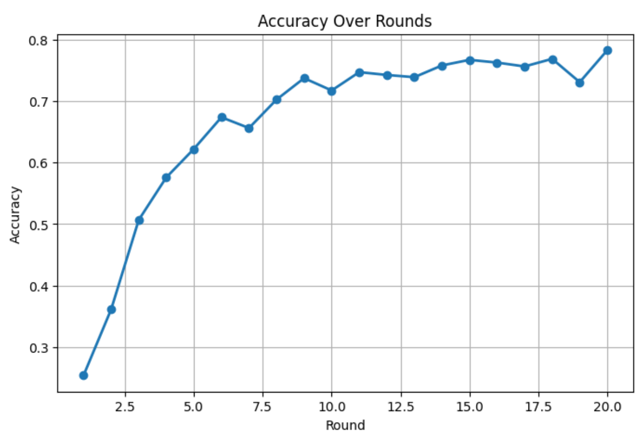

Figure 5: Ideal Scenario for Scale-Free

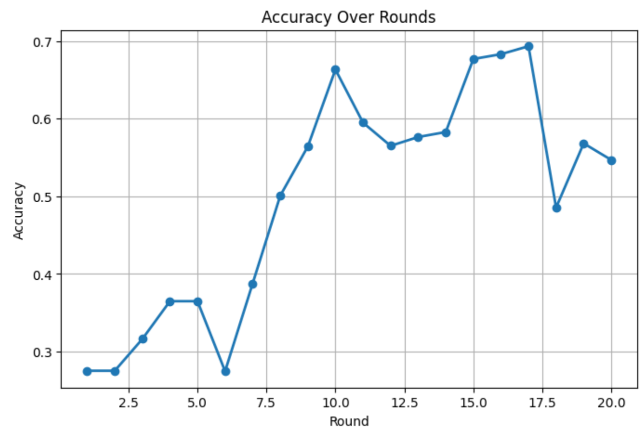

Figure 6: Naive Approach for Scale-Free

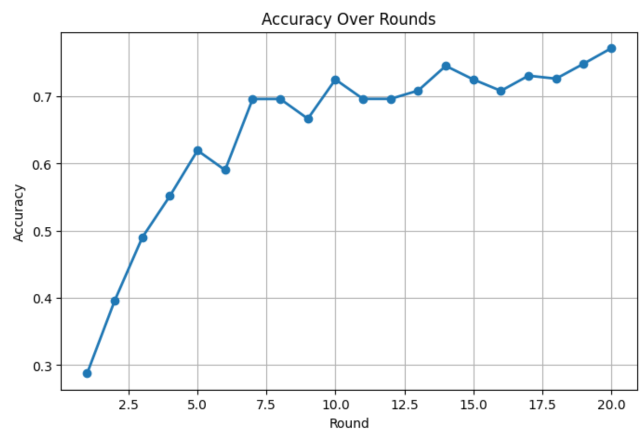

Figure 7: Dynamic Recovery for Scale-Free

When comparing the Recovery performance across the two network structures, the results demonstrate that the rerouting algorithm is highly effective regardless of topology. In the Erdős–Rényi graph, where connectivity is randomly distributed, the Recovery protocol achieved an accuracy of 77.69%, recovering a significant portion of the Ideal scenario's 80.25% performance. Similarly, in the Scale-Free topology, which is characterized by central 'hub' nodes, the Recovery protocol reached 77.17% accuracy, restoring performance remarkably close to the Ideal case of 78.33% and vastly outperforming the Naive approach's 54.69%. Contrary to the expectation that hub failures might severely hamper connectivity, the Scale-Free topology actually exhibited a smaller relative performance drop (1.16%) compared to the Erdős–Rényi graph (2.56%). This suggests that the dynamic routing algorithm successfully exploited the high interconnectivity of the remaining hubs to maintain robust model convergence. Ultimately, the consistency across both topologies confirms that the proposed recovery mechanism generalizes well, reliably mitigating network disturbances in both random and power-law distributed networks.

Results for ER Topology with Different Random Seeds: To test robustness, we ran an additional experiment on the same Erdős–Rényi (ER) topology using different random seeds. This lets us evaluate how recovery performance varies across network realizations. The metrics for the three scenarios are shown below.

- Ideal Scenario: Achieved an accuracy of 79.50%, which is highly like the ER baseline. This is shown in Figure 8.
- Naive Approach: Performance dropped significantly to 61.96%. This is shown in Figure 9.
- Recovery Protocol: 77.27% accuracy which is close to the Ideal Scenario. This is shown in Figure 10.

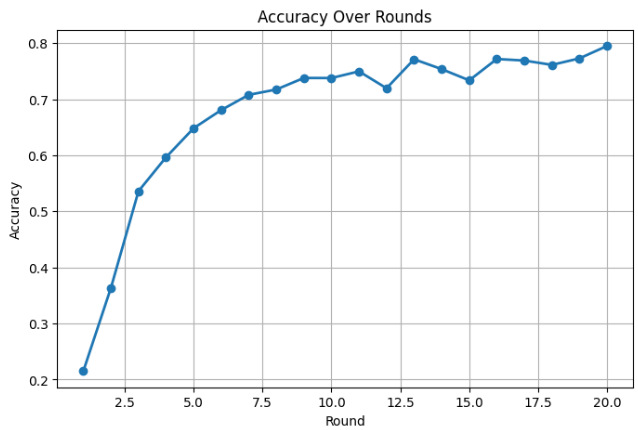

Figure 8: Ideal Scenario for Different Seed

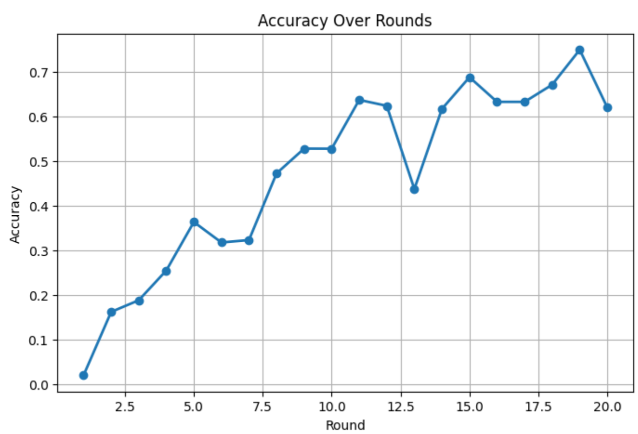

Figure 9: Naive Approach for Different Seed

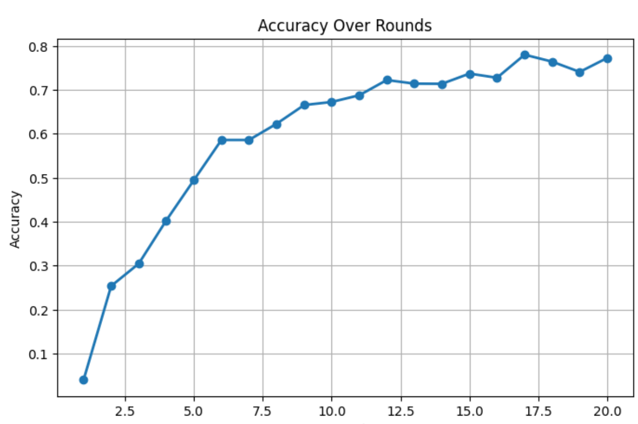

Figure 10: Dynamic Recovery for Different Seed

When analyzing the performance across different random seeds, the results demonstrate a high degree of statistical consistency. Despite the specific adjacency matrices and link locations changing between the two experiments, the Recovery protocol achieved accuracy levels nearly identical to the initial ER trial. This indicates that the rerouting algorithm's success is not an artifact of a specific favorable network initialization, but rather a robust property of the protocol itself. The Naive approach consistently fails to cope with disturbances regardless of the seed, while the Recovery mechanism reliably leverages the uniform connectivity of the ER model to find alternative paths. This reproducibility confirms that the framework effectively generalizes across different realizations of random networks without significant variance in convergence or stability.

While the proposed dynamic re-routing framework successfully enhances the resilience of Federated Learning in unreliable networks, it introduces specific limitations that merit further investigation. The primary trade-off is increased latency; utilizing multi-hop pathways inevitably incurs higher transmission delays compared to direct server communication. This latency is compounded by the computational overhead of executing Dijkstra’s algorithm on edge devices, which may strain resource-constrained hardware during path recalculation. Furthermore, routing model updates through neighbor peers introduces privacy and security vulnerabilities, as intermediate nodes could theoretically inspect or tamper with gradient information during transit, necessitating robust encryption layers that might further increase overhead.

To address these challenges, future extensions of this work will explore alternative Federated Learning frameworks beyond FedAvg. Algorithms such as FedProx or SCAFFOLD could be integrated to better handle the statistical heterogeneity and "straggler" effects caused by variable routing delays. Additionally, the reactive routing protocol could be upgraded to a predictive Reinforcement Learning (RL) model, allowing devices to anticipate link failures and switch paths preemptively rather than reacting after a disconnect occurs. Finally, transitioning from a synchronous to an asynchronous aggregation protocol would allow the server to incorporate updates as they arrive, thereby mitigating the idle time caused by long multi-hop traversals and improving overall convergence speed in highly volatile environments.

5. Conclusion

This project demonstrates that Federated Learning can effectively maintain high performance in unstable edge environments through the integration of dynamic re-routing protocols. By moving beyond standard "drop-out" mechanisms, our framework utilizes a multi-hop recovery algorithm to salvage model updates that would otherwise be lost to network disturbances. Experimental results across both Erdős–Rényi and Scale-Free topologies confirm the efficacy of this approach; while naive implementations suffered accuracy degradations of over 20% under stochastic failure conditions, our recovery protocol successfully restored model accuracy to near-ideal levels (~77-78%). Crucially, additional validation using varying random seeds on the Erdős–Rényi topology demonstrated consistent recovery performance across different network realizations. This confirms that the protocol's resilience is statistically robust and not merely an artifact of a specific favorable graph initialization. These findings validate that active path reconstruction is a viable strategy for preserving the statistical integrity of the global model without requiring a flawless communication infrastructure.

The potential impact of this work extends significantly into the deployment of resilient Edge AI. By decoupling training stability from network reliability, this framework paves the way for Federated Learning to be utilized in harsh or resource-constrained environments, such as disaster response zones, rural IoT networks, or congested industrial settings. Ultimately, this shifts the paradigm of distributed learning from one that requires rigid connectivity to one that adapts fluidly to the chaotic nature of real-world wireless networks.

6. Contributions

The development of this simulation framework was a collaborative effort with distinct areas of technical focus. Nikhil took primary responsibility for the simulation environment, implementing the baseline Federated Averaging (FedAvg) framework and the DisturbanceManager to model stochastic network failures and device dropouts. Ronak led the development of the resilience mechanisms, specifically designing and coding the dynamic re-routing algorithm that enables multi-hop traversal using Dijkstra's algorithm. Both team members collaborated extensively on the high-level architectural design of these algorithms and worked jointly on the rigorous hyperparameter tuning required to optimize model convergence under non-IID data distributions.

From a software engineering perspective, this project highlighted the critical importance of decoupling state management from experimental variables. We learned that separating the static base network state from ephemeral round-to-round disturbances—rather than mutating the graph directly—allows for stateless, reproducible simulations. This was complemented using abstraction layers for routing strategies, which enabled us to seamlessly toggle between "naive" BFS approaches and "dynamic" weighted algorithms without refactoring the core device logic.

Furthermore, the project reinforced the value of "configuration-as-code" and upfront resource accounting. By implementing dataclass-based configurations and building battery and byte tracking into the device objects from the start, we were able to conduct complex parameter sweeps and introduce disturbances without breaking existing functionality. This modular approach, combined with backward-compatible metrics logging, demonstrated that designing extensibility and reproducibility is just as vital as the algorithmic logic itself in research-driven software.

REFERENCES

[1] Hosseinalipour, Seyyedali, et al. "From federated to fog learning: Distributed machine learning over heterogeneous wireless networks." IEEE Communications Magazine 58.12 (2020): 41-47.

[2] L. U. Khan, M. Alsenwi, Z. Han and C. S. Hong, "Self Organizing Federated Learning Over Wireless Networks: A Socially Aware Clustering Approach," Organizing Federated Learning Over Wireless Networks: A Socially Aware Clustering Approach," 2020 International Conference on Information Networking (ICOIN), Barcelona, Spain, 2020, pp. 453-458, doi: 10.1109/ICOIN48656.2020.9016505.

[3] Kim, Yeongwoo, et al. "Dynamic clustering in federated learning" ICC 2021-IEEE International Conference on Communications. IEEE, 2021.

[4] K. Bonawitz et al., "Towards federated learning at scale: System design," in Proceedings of Machine Learning and Systems (SysML), Palo Alto, CA, USA, 2019

[5] https://www.kaggle.com/datasets/hojjatk/mnist-dataset